Secrets of Successful Project Management

Karl E. Wiegers

Process Impact
www.processimpact.com

Managing software projects is difficult under the best circumstances. Unfortunately, many new project managers receive virtually no job training. Sometimes you must rely on coaching and survival tips from people who have already done their tour of duty in the project management trenches. Here are 20 such tips for success, which I've learned from both well-managed and challenged projects. Keep these suggestions in mind during your next project, recognizing that none of them is a silver bullet for your project management problems.

Laying the Groundwork

Tip #1: Define project success criteria. At the beginning of the project, make sure the stakeholders share a common understanding of how they will determine whether this project is successful. Too often, meeting a predetermined schedule is the only apparent success factor, but there are certainly others. Some examples are increasing market share, reaching a specified sales volume or revenue, achieving specific customer satisfaction measures, retiring a high-maintenance legacy system, and achieving a particular transaction processing volume and correctness.

Tip #2: Identify project drivers, constraints, and degrees of freedom. Every project needs to balance its functionality, staffing, budget, schedule, and quality objectives. Define each of these five project dimensions as either a constraint within which you must operate, a driver aligned with project success, or a degree of freedom that you can adjust within some stated bounds to succeed. For more details about this, see Chapter 2 of my *Creating a Software Engineering Culture* (Dorset House, 1996).

Tip #3: Define product release criteria. Early in the project, decide what criteria will determine whether or not the product is ready for release. You might base release criteria on the number of high-priority defects still open, performance measurements, specific functionality being fully operational, or other indicators that the project has met its goals. Whatever criteria you choose should be realistic, measurable, documented, and aligned with what "quality" means to your customers.

Tip #4: Negotiate commitments. Despite pressure to promise the impossible, never make a commitment you know you can't keep. Engage in good-faith negotiations with customers and managers about what is realistically achievable. Any data you have from previous projects will help you make persuasive arguments, although there is no real defense against unreasonable people.

Planning the Work

Tip #5: Write a plan. Some people believe the time spent writing a plan could be better spent writing code, but I don't agree. The hard part isn't writing the plan. The hard part is actually doing the planning—thinking, negotiating, balancing, talking, asking, and listening. The time you spend analyzing what it will take to solve the problem will reduce the number of surprises you have to cope with later in the project.

Tip #6: Decompose tasks to inch-pebble granularity. Inch-pebbles are miniature milestones. Breaking large tasks into multiple small tasks helps you estimate them more accurately, reveals work activities you might not have thought of otherwise, and permits more accurate, fine-grained status tracking.

Tip #7: Develop planning worksheets for common large tasks. If your team frequently undertakes certain common tasks, such as implementing a new object class, develop activity checklists and planning worksheets for these tasks. Each checklist should include all of the steps the large task might need. These checklists and worksheets will help each team member identify and estimate the effort associated with each instance of the large task he or she must tackle.

Tip #8: Plan to do rework after a quality control activity. Almost all quality control activities, such as testing and technical reviews, find defects or other improvement opportunities. Your project schedule or work breakdown structure should include rework as a discrete task after every quality control activity. If you don't actually have to do any rework, great; you're ahead of schedule on that task. But don't count on it.

Tip #9: Plan time for process improvement. Your team members are already swamped with their current project assignments, but if you want the group to rise to a higher plane of software engineering capability, you'll have to invest some time in process improvement. Set aside some time from your project schedule, because software project activities should include making process changes that will help your next project be even more successful. Don't allocate 100% of your team members' available time to project tasks and then wonder why they don't make any progress on the improvement initiatives.

Tip #10: Manage project risks. If you don't identify and control risks , they will control you. Spend some time during project planning to brainstorm possible risk factors, evaluate their potential threat, and decide how you can mitigate or prevent them. For a concise tutorial on software risk management, see my article "Know Your Enemy: Software Risk Management" (Oct. 1998).

Estimating the Project

Tip #11: Estimate based on effort, not calendar time. People generally provide estimates in units of calendar time, but I prefer to estimate the amount of effort (in labor-hours) associated with a task, then translate the effort into a calendar-time estimate. This translation is based on estimates of how many effective hours I can spend on project tasks per day, any interruptions or emergency bug fix requests I might get, meetings, and all the other places into which time disappears.

Tip #12: Don't schedule people for more than 80%of their time. Tracking the average weekly hours that your team members actually spend working on their project assignments is a real eye-opener. The task-switching overhead associated with the many activities we are all asked to do reduces our effectiveness significantly. Don't assume that just because someone spends 10 hours per week on a particular activity, he or she can do four of them at once; you'll be lucky if he or she can handle three.

Tip #13: Build training time into the schedule. Determine how much time your team members typically spend on training activities annually, and subtract that from the time available for them to be assigned to project tasks. You probably already subtract out average values for vacation time, sick time, and other assignments; treat training time the same way.

Tip #14: Record estimates and how you derived them. When you prepare estimates for your work, write down those estimates and document how you arrived at each of them. Understanding the assumptions and approaches used to create an estimate will make them easier to defend and adjust when necessary, and it will help you improve your estimation process.

Tip #15: Use estimation tools. Many commercial tools are available to help you estimate entire projects. With their large databases of actual project experience, these tools can give you a spectrum of possible schedule and staff allocation options. They'll also help you stay out of the "impossible region," combinations of product size, team size, and schedule where no known project has been successful. A good tool to try is Estimate Pro from the Software Productivity Centre (www.spc.ca).

Tip #16: Respect the learning curve. If you're trying new processes, tools, or technologies for the first time on this project, recognize that you will pay a price in terms of a short-term productivity loss. Don't expect to get the fabulous benefits of new software engineering approaches on the first try, so build extra time into the schedule to account for the inevitable learning curve.

Tip #17: Plan contingency buffers. Things never go precisely as you plan on a project, so your budget and schedule should include some contingency buffers at the end of major phases to accommodate the unforeseen. Unfortunately, your manager or customer may view these buffers as padding, rather than the sensible acknowledgement of reality that they are. Point to unpleasant surprises on previous projects as a rationale for your foresight.

Tracking Your Progress

Tip #18: Record actuals and estimates. If you don't record the actual effort or time spent on each task and compare them to your estimates, you'll never improve your estimating approach. Your estimates will forever remain guesses.

Tip #19: Count tasks as complete only when they're 100% complete. One benefit of using inch-pebbles for task planning is that you can classify each small task as either done or not done, which is more realistic than trying to estimate what percent of a large task is complete at any time. Don't let people "round up" their task completion status; use explicit criteria to tell whether a step truly is completed.

Tip #20: Track project status openly and honestly. Create a climate in which team members feel safe reporting project status accurately. Strive to run the project from a foundation of accurate, data-based facts, rather than from the misleading optimism that sometimes arises from fear of reporting bad news. Use project status information to take corrective actions when necessary and to celebrate when you can.